Metrix User Guide. Error Estimates and Mesh Control for Anisotropic Mesh Adaptation

Frédéric Alauzet, Adrien Loseille

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

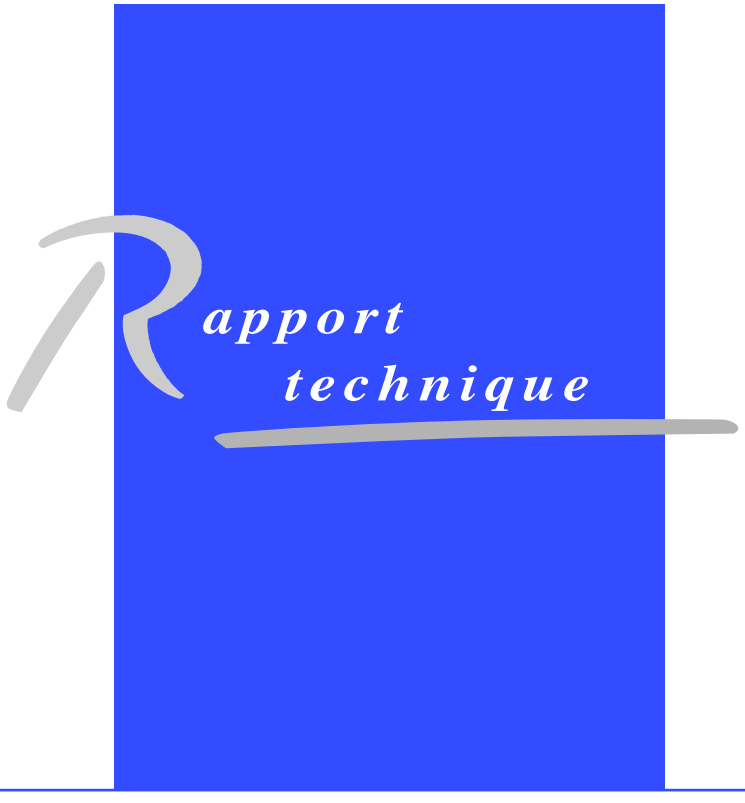
Metrix *User Guide. Error Estimates and Mesh Control for Anisotropic Mesh Adaptation*

Frédéric Alauzet and Adrien Loseille

N° 0363

February 16, 2009

Thème NUM



ISSN 0249-0803 ISRN INRIA/RT--0363--FR+ENG

`Metrix` User Guide. Error Estimates and Mesh Control for Anisotropic Mesh Adaptation

Frédéric Alauzet* and Adrien Loseille†

Thème NUM — Systèmes numériques
Projet Gamma

Rapport technique n° 0363 — February 16, 2009 — 36 pages

Abstract: This technical note describes the main features of `Metrix`‡, a software that computes metric field from *a priori* error estimates or meshes and that handles several metric operations. The aim of the software, the input and the output files, the list of modules and options are given in this document. References that explain the various possibilities of the code are provided.

`Metrix` has been developed within the GAMMA research project at INRIA Paris-Rocquencourt.

This document describes the features of the current version: release `V1.2` (January 2009).

Key-words: Metric, error estimate, interpolation error, anisotropic mesh adaptation.

* Email : Frederic.Alauzet@inria.fr
† Email : Adrien.Loseille@inria.fr
‡ This software was registered with the APP under n° IDDN.FR.001.070014.000.S.P. 2009.000.10000 on february 10, 2009.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

`Metrix` Guide de l'utilisateur. Estimateurs d'erreur et contrôle de maillage pour l'adaptation de maillage anisotrope

Résumé : Ce rapport technique décrit les principales fonctions de `Metrix`[§], un logiciel qui calcule des champs de métriques à partir d'estimateurs d'erreur ou de maillages et qui permet d'appliquer plusieurs opérateurs sur les métriques. Les fonctionnalités du logiciel, les formats des fichiers d'entrée/sortie, la liste des modules et des options sont donnés dans ce document. Les références qui décrivent les différentes possibilités offertes par le code sont fournis.

`Metrix` a été développé au sein de l'équipe projet GAMMA à l'INRIA Paris-Rocquencourt.

Ce document décrit les fonctionnalités de la version courante : version `V1.2` (Janvier 2009).

Mots-clés : Métrique, estimateur d'erreur, erreur d'interpolation, adaptation de maillage anisotrope.

[§] Ce logiciel a été enregistré à l'APP sous le numéro n° IDDN.FR.001.070014.000.S.P. 2009.000.10000 le 10 février 2009.

Contents

1 Metrix overview

1.1 Context of metric-based mesh adaptation

Mesh adaptation provides a way to control the accuracy of the numerical solution by modifying the domain discretization according to size and directional constraints. When dealing with real life flow problems, Hessian based unstructured mesh adaptation has already proved its efficiency to improve the ratio between the solution accuracy and the number of degrees of freedom (the problem complexity).

The generation of anisotropic adapted meshes uses the notion of length in a metric space. The idea is to introduce a metric tensor in the dot product definition to modify size evaluation in all directions. A metric is a $n \times n$ symmetric definite positive matrix, where n is the space dimension. The mesh is automatically adapted by generating *a unit mesh* with respect to this metric, *i.e.,* the mesh is such that all edges have a length close to one in the metric and such that all elements are almost regular. The mesh is then uniform in the prescribed metric space and, non-uniform and anisotropic in the Euclidean space.

1.2 Main features

`Metrix` is a software that provides by various ways metric to govern the mesh generation. Generally, these metrics are constructed from error estimates (a priori or a posteriori) applied to the numerical solution. `Metrix` computes metric fields from scalar solutions by means of several error estimates: interpolation error, iso-lines error estimate, interface error estimate and goal oriented error estimate. It also contains several modules that handle meshes and metrics. For instance, it extracts the metric associated with a given mesh and it performs some metric operations such as: metric gradation and metric intersection.

1.3 Languages and platforms

The program is entirely written in C (C89 ANSI norm). The current version consists of about 40 000 lines of optimized source code. The code is free of any external libraries. Hence, the code is very portable and has been successfully compiled and tested on all major computer architectures (i.e., HP, IBM, Intel- based PC, etc.) and operating systems (Unix/Linux, WindowsNT, Mac OS).

1.4 Software integration

If `Metrix` is integrated in a software package, only the input and output routines need to be modified, for efficiency and compatibility purposes. In this context, no more than a few routines need to be modified and adapted to support the user file formats.

1.5 Distribution

An evaluation copy of Metrix software for a limited period of time can be obtained by contacting the authors at INRIA :

Frédéric ALAUZET
INRIA, Domaine de Voluceau
BP 105, 78153 Le Chesnay cedex, France
Email: frederic.alauzet@inria.fr

Adrien LOSEILLE
INRIA, Domaine de Voluceau
BP 105, 78153 Le Chesnay cedex, France
Email: adrien.loseille@inria.fr

2 Input and output data

`Metrix` requires the specification of meshes, solution fields and metric fields. It ouputs metric fields and derivatives fields. The specification of the discrete support, *i.e.* the mesh, is done by the `mesh` format. 2D meshes, 3D surface meshes and 3D meshes can be specified. As regards general fields, they are specified with the `sol` format.

2.1 Mesh specification

Meshes are described using the `mesh` file format. The `mesh` format describes precisely meshes and also the surface features. This format is composed of a single (ASCII or binary) file, `xxx.mesh` or `xxx.meshb`. This file contains all the information needed to describe entirely the mesh.

Its structure is organized as a series of fields identified by keywords. The blanks, "newline" or <CR> and tabs are considered as item separators. A comment line starts with the character # and ends at the end of the line. The comments are placed exclusively between the fields. The mesh file must start with the descriptor :

```
MeshVersionFormatted 2
Dimension 3 # or 2 in 2D
```

The other required fields for `Metrix` correspond to the geometry (*i.e.,* the coordinates) and to the topology description (*i.e.,* the mesh entities). In the following tables, the term v_i indicates a vertex index, *i.e.,* the ith vertex in the vertices list. The vertices are defined by their coordinates either in simple or in double precision. The reference is an integer attached to the vertex. For instance, it can represent a tag for boundary conditions or the tag of a partition. The elements inside the domain or on the boundary are defined by their list of vertices where each vertex id is given thanks to an integer. The reference is an integer attached to the element.

3D meshes:

Keyword	Card.	Syntax	Range
`Vertices`	nv	$x_i\ y_i\ z_i\ vref_i$	$\{i = 1, nv\}$
`Tetrahedra`	nt	$v_i^1\ v_i^2\ v_i^3\ v_i^4\ tref_i$	$\{i = 1, nt\}$
`Triangles`	nf	$v_i^1\ v_i^2\ v_i^3\ fref_i$	$\{i = 1, nf\}$

2D meshes:

Keyword	Card.	Syntax	Range
`Vertices`	nv	$x_i\ y_i\ vref_i$	$\{i = 1, nv\}$
`Triangles`	nt	$v_i^1\ v_i^2\ v_i^3\ tref_i$	$\{i = 1, nt\}$
`Edges`	ne	$v_i^1\ v_i^2\ eref_i$	$\{i = 1, ne\}$

Finally, the data structure must end with the keyword:

```
End
```

Let us give an example:

```
MeshVersionFormatted 2
Dimension 2

# Set of mesh vertices (x,y,ref)
Vertices
581
0.1 1.  0
0.333 12.125 0
.........

# Set of mesh triangles (v1,v2,v3,ref)
Triangles
1162
1 28 521 0
23 45 77 0
.........

# Set of mesh edges (v1,v2,ref)
Edges
212
1 28 3
28 34 3
.........

End
```

2.2 Solution or metric specification

Fields or solutions are described using the `sol` file format. The `sol` format describes several types of solutions (scalar, vector, tensors,...) which can be defined on different mesh entities. This format is composed of a single (ASCII or binary) file, `xxx.sol` or `xxx.solb`.

Its structure is organized as a series of fields identified by keywords. The blanks, "newline" or <CR> and tabs are considered as item separators. A comment line starts with the character # and ends at the end of the line. The comments are placed exclusively between the fields. The sol file must start with the descriptor :

```
MeshVersionFormatted 2
Dimension 3 # or 2 in 2D
```

The solutions fields for `Metrix` are associated with the vertices of the given mesh and are

defined by the keyword `SolAtVertices`. This keyword is followed by the number of entities (here vertices) supporting the solution, the number of types and the list of solution types. The type of solutions handled by `Metrix` can be scalar, vector or tensor fields (for instance a metric). They are defined as follow depending on the dimension

3D solution type:

Field	Type	Syntax	Range
`Scalar`	1	f_i	{i=1,nv}
`Vector`	2	$f_i^1\ f_i^2\ f_i^3$	{i=1,nv}
`Tensor`	3	$f_i^1\ f_i^2\ f_i^3\ f_i^4\ f_i^5\ f_i^6$	{i=1,nv}

2D solution type:

Field	Type	Syntax	Range
`Scalar`	1	f_i	{i=1,nv}
`Vector`	2	$f_i^1\ f_i^2$	{i=1,nv}
`Tensor`	3	$f_i^1\ f_i^2\ f_i^3$	{i=1,nv}

where the convention for tensors is

$$\mathcal{M}_{2D} = \begin{pmatrix} f_i^1 & f_i^2 \\ f_i^2 & f_i^3 \end{pmatrix} \qquad \text{and} \qquad \mathcal{M}_{3D} = \begin{pmatrix} f_i^1 & f_i^2 & f_i^4 \\ f_i^2 & f_i^3 & f_i^5 \\ f_i^4 & f_i^5 & f_i^6 \end{pmatrix}$$

Finally, the data structure must end with the keyword:

```
End
```

Let us given an example for each solution type:

Scalar

```
MeshVersionFormatted 2

Dimension 2

SolAtVertices
581
1 1
1.
0.125
0.125
.....
End
```

Vector

```
MeshVersionFormatted 2

Dimension 2

SolAtVertices
581
1 2
0. 0.
0.3945 2.55264e-05
0.245741 1.14493e-05
.....
End
```

Tensor

```
MeshVersionFormatted 2

Dimension 2

SolAtVertices
581
1 3
70.8852 0 70.8852
72.6135 0 72.6135
63.7954 0 63.7954
.....
End
```

and a final example for several solution types associated to mesh vertices. There are three solution types. The first type is a scalar, the second a vector and the third one a scalar. In the file, the first column corresponds to the first scalar solution field. The second and the third columns correspond to the vector solution field. And the last column is associated with to the second scalar solution field.

```
MeshVersionFormatted
2

Dimension
2

SolAtVertices
25282
3 1 2 1
1 0 0 2.5
0.125 0 0 0.25
0.125 0 0 0.25
1 0 0 2.5
0.425971 0.3945 2.55264e-05 0.941469
............
End
```

3 Metrix modules description

This section presents all the Metrix modules and the options that can be prescribed with the command line for each module. Notably, we specify the input and the output files for each module. We also provide some generic options for the I/O that are only accessible with the command line. Some advanced options are only accessible with the Metrix data file. All these options are given in Section 4.2.

Be careful, the parser of the data file is case sensitive for the keywords. For instance, -Hmet is not equivalent to -hMet.

Metrix is composed of the main module that computes a metric field from a given solution and of several modules to handle metric fields. The list of all the modules is:

-O 1 compute a metric field from a solution thanks to the chosen error estimate

-O 3 compute the metric intersection of several metrics

-O 4 compute the natural metric of a given mesh

-O 5 gradation of a given metric field

-O 7 compute a user metric field thanks to some parameters specified on the boundary

-O 10 compute the log or the exp of a given metric field

3.1 Module 1: Metric from an error estimate

This is the main module of Metrix in which a metric is derived from an error estimate. The following error estimates are available:

- the interpolation error in $\mathbf{L}^p$-norm, *i.e.,* for multi-scales mesh adaptation, see [7] for details
- an error estimate following the iso-lines (the level set) of a solution field, see [6] for details
- an error estimate dedicated to an interface (for multi-fluid problems), see [6] for details
- a goal oriented error estimate, see [7] for details.

When this module is called with the command line, then all fields of the solution file are taken into account, that is to say a metric is derived for each field and the final resulting metric is the intersection of all these metrics. Moreover, all the specified parameters are applied for all fields.

If the user wants to select some of the fields and to specify particular parameters for each field, then the user must use the data file, see Section 4.2. For instance, the use of the data file is mandatory if the user wants to specify a control of the interpolation error for one field and the interface error estimate for another field.

Moreover, the goal oriented error estimate can only be activated with the data file. It cannot be called with the command line.

The **input and output files** are selected with:

`-in [char]`	Specify the input file name for the mesh and for the solution.
`-sol [char]`	Specify the solution input file name, if it is different from the mesh file name.
`-out [char]`	Specify the metric output file name. If not specified, the output file name is the input file name concatenated with "`.o`".

The different **error estimates** are specified with

`-Hmet`	Interpolation error estimate.
`-Gmet`	Iso-lines error estimate.
`-Lmet`	Interface error estimate.

And the associated **generic options** are:

`-C [int]`	Specify the mesh complexity. The theoretical continuous equivalent of the number of vertices. Cannot be used with `-E [float]`.
`-E [float]`	Specify the error threshold. It is mandatory for unsteady problems for consistency reasons. Cannot be used with `-C [int]`.

The other possible options for the command line are `-iso`, `-hmin [float]`, `-hmax [float]` and `-Cmax [int]` that are presented in the generic options section (3.7).

The possible parameters for the **interpolation error estimate** are:

`-L [int]`	Control the interpolation error in $\mathbf{L}^p$-norm where $p \geq 1$ is given by `[int]`.
`-L inf`	Control the interpolation error in $\mathbf{L}^\infty$-norm.
`-green`	Green formula based hessian reconstruction. Advised for steady simulations.
`-L2p`	$\mathbf{L}^2$-projection based hessian reconstruction. Advised for unsteady simulations. This is the default value for the hessian reconstruction.

The possible parameters for the **interface error estimate** are:

`-hls [float]`	Specify the mesh size normal to the interface.
`-tls [float]`	Specify the thickness of the interface refined region.

Let us give two examples:

```
metrix -O 1 -in naca -out metric -Hmet -L 2 -green -C 5000
```

here `Metrix` computes the metric with a control of the interpolation error in $\mathbf{L}^2$-norm. The green formula is used for the Hessian reconstruction. A mesh complexity of 5 000 has been

specified, it represents the error threshold. It reads the mesh file `naca.mesh[b]`, and the solution file `naca.sol[b]`. The output metric is written in the file `metric.sol[b]`.

```
metrix -O 1 -in dam -out met -Lmet -L 2 -E 0.1 -hls 0.01 -tls 0.05
```

here `Metrix` computes the metric for a mesh adaptation around an interface. The mesh size in the tangent plane to the interface is controlled with the error level equal to 0.1. The mesh size normal to the interface is set to 0.01. The adapted regions have a thickness of 0.05 on both sides of the interface. `Metrix` reads the mesh file `dam.mesh[b]`, and the solution file `dam.sol[b]`. The output metric is written in the file `met.sol[b]`.

Other modules can be called inside this module. Module 1 can be combined with the metric gradation module (5) and the user metric module (7).

The gradation module becomes automatically active if the option `-hgrad [float]` is specified. Other options related to mesh gradation can be prescribed. These options are defined in the module 5 section. Mesh gradation can also be activated with the data file.

The user metric becomes automatically active if the option `-bdy` is specified. The other options related to the user metric can be prescribed. These options are defined in the module 7 section. The user metric can also be activated thanks to the data file.

3.2 Module 3: metrics intersection

This module reads a series of metrics that are intersected into a single one, see [1] for technical details.

The following options are required for this module:

`-in [char]`	Specify the input base name of the list of metric.
`-out [char]`	Specify the metric output file name. If not specified, the output file name is the root input file name concatenated with "`.o`".
`-nbm [int]`	The number of metrics to intersect.

`Metrix` reads the list of files $\{\texttt{rootname}.i\texttt{.sol[b]}\}_{i=0\ldots n-1}$ where `rootname` is the input base name and n is the number of metrics to intersect.

Let us give an example:

```
metrix -O 3 -in met -out finalmet -nbm 5
```

here `Metrix` reads the list of files `met.0.sol[b]`, `met.1.sol[b]`, ..., `met.4.sol[b]` and it writes `finalmet.sol[b]`.

3.3 Module 4: metric of a mesh

This module computes the natural metric of a given mesh. More precisely, the size of the elements belonging to the ball of each vertex is analyzed. Then, the metric which fits best these sizes is computed. This is done by a $\mathbf{L}^2$-projection operator that has been extended to metric fields, see [7] for technical details. The histogram of the mesh quality related to its natural metric is printed.

The following options are required for this module:

`-in [char]`	Specify the mesh input file name.
`-out [char]`	Specify the metric output file name. If not specified, the output file name is the input file name concatenated with "`.o`".

Let us give an example:

```
metrix -O 4 -in carre
```

here `Metrix` reads the mesh file `carre.mesh[b]` and writes the metric file `carre.o.sol[b]`.

3.4 Module 5: metric gradation

The metric gradation is a regularization procedure on a metric field enabling to:

1. smooth the metric field. The new gradated metric field has some regularity properties
2. control the metric size variation, that is to say, the difference of size prescription in each direction between two neighboring vertices is bounded by a given "gradient" value.

This procedure is usually mandatory as in real-life applications, the metric field issued from an error estimate is generally not regular. Indeed, the solution field on which the metric is based is non-regular and can contain discontinuities.

The mesh gradation module reads a metric field defined on a mesh, applies the metric gradation algorithm (see [2] for details) and outputs the new graded metric. Several methods have been proposed in [2]. They are accessible with the `Metrix` data file, Section 4.2. Only the following options are accessible with the command line:

`-in [char]`	Specify the input file name for the mesh and for the metric.
`-sol [char]`	Specify the metric input file name, if it is different from the mesh file name.
`-out [char]`	Specify the metric output file name. If not specified, the output file name is the input file name concatenated with "`.o`".
`-hgrad [float]`	Maximal size variation coefficient between two neighboring vertices. This coefficient must be greater than 1.
`-tgrad [int]`	Metric gradation type. Default value is 0. =0 the gradation is homogeneous in the physical space. =1 the gradation is homogeneous in the metric space. =2 the gradation is homogeneous in a mixed physical/metric space defined by `cgrad`.
`-cgrad [float]`	Coefficient that defined the mixed physical/metric space. This coefficient lies in $[0, 1]$. Default value is 0.125.

By default, the mesh gradation algorithm is performed edge by edge. A super-edge algorithm is available with the data file.

As regards the size variation law, the default law is the h-linear law. Other variation laws can be chosen the data file: the h-geometric law, the λ-linear law and the h^{-1}-linear law. But, changing the variation law has a negligible impact on the edge by edge gradation algorithm as, in this case, they almost reduce to the linear law.

Let us give an example:

```
metrix -O 5 -in carre -out metgrad -hgrad 2 -tgrad 2 -cgrad 0.5
```

here `Metrix` reads the mesh file `carre.mesh[b]` and the metric file `carre.sol[b]` and writes the graded metric file `metgrad.sol[b]`.

3.5 Module 7: user metric

The user metric enables the user to specify some parameters in order to govern the generation of a mesh. From the characteristic of the surface mesh and these parameters, a metric field is defined in the whole domain to govern the mesh generation. Usually, this module is used to set up an adequate pseudo-uniform mesh. More details are given in [2].

First, the metric of the surface mesh is defined by a $\mathbf{L}^2$-projection. This metric field is defined in the tangent plane of the surface. The size in the normal direction is defined by the user either by a specified size or a coefficient of the minimal local surface mesh size. At this point, a 3D metric at each vertex of the surface is defined. Finally, this metric field is propagated in the volume thanks to the mesh gradation which controls the metric growth from the surface into the volume, see the gradation module (Section 3.4) for details.

The following options are used for this module:

`-in [char]`	Specify the mesh input file name.
`-out [char]`	Specify the metric output file name. If not specified, the output file name is the input file name concatenated with "`.o`".
`-coef [float]`	Specify the normal size coefficient. The normal size at each vertex is the minimal size in the tangent plane multiplied by this coefficient. Default value is 1. Cannot be used with `-hnor`.
`-hnor [float]`	Specify the normal size of the surface metric. Cannot be used with `-coef`.

These options have to be completed with the mesh gradation options.

Let us give an example:

```
metrix -O 7 -in carre -out init -coef 2 -hgrad 2 -tgrad 1
```

here `Metrix` reads the mesh file `carre.mesh[b]` and writes the metric file `init.sol[b]`.

3.6 Module 10: operations on metrics

This module performs a series of operations on a metric field. At present, only the exponential and the logarithmic operations are available. These operations are recurrent in the Log-Euclidean framework, see [7] for technical details.

The following options are required for this module:

`-in [char]`	Specify the input file name for the mesh and for the metric.
`-out [char]`	Specify the metric output file name. If not specified, the output file name is the input file name concatenated with "`.o`".
`-log`	Compute the logarithm of the given metric.
`-exp`	Compute the exponential of the given metric.

Let us give an example:

```
metrix -O 10 -in met -out logmet -log
```

here `Metrix` reads the mesh file `met.mesh[b]` and the metric file `met.sol[b]`. Then, it computes the logarithmic metric and writes `logmet.sol[b]`.

3.7 Generic options

The list of options is accessible by calling the inline help with the syntax: `metrix -h`.

The input and the output files names are given with the following option:

`-in [char]`	Specify the input file name for the mesh and the solution and/or the metric.
`-sol [char]`	Specify the solution input file name, if it is different from the mesh file name.
`-out [char]`	Specify the metric output file name. If not specified, the output file name is the input file name concatenated with "`.o`".

The following options are standard options used to define the metric:

`-iso`	Output an isotropic metric. The computed anisotropic metric is transformed into an isotropic one.
`-hmin [float]`	Specify the minimal allowable size prescribed by the metric.
`-hmax [float]`	Specify the maximal allowable size prescribed by the metric.
`-Cmax [int]`	Specify the maximal complexity authorized for the final metric.

We give here the list of options related to the I/O that are only accessible with the command line:

`-f`	Save output files in ascii. By default, `Metrix` writes the output files in binary format.
`-f32`	Save output files in 32-bits real number. By default, `Metrix` writes in 64-bits (double precision).
`-f64`	Force `Metrix` to save output files in 64-bits real number.
`-h`	Print the help in the terminal.
`-o [int]`	Tune the files output level.
`-v [int]`	Tune the level of verbosity
`-w`	Write a generic option file.

4 Metrix usage

4.1 Command line

To execute `Metrix` the following syntax is used

```
metrix -O module -in filein[.mesh[b]] [-sol filesol[.sol[b]]]
       [-out fileout[.sol[b]]] [list of options...]
```

`Metrix` reads the following input files:

- the mesh `filein.mesh[b]`
- the data file `filein.metrix` or `DEFAULT.metrix` if the first one is not found
- if a solution or a metric file is required, the file `filesol.sol[b]` (or `filein.sol[b]` if the option `-sol [char]` is not used) is read

and it writes the following output files:

- the computed metric `fileout.sol[b]` (or `filein.o.sol[b]` if the option `-out [char]` is not used
- derivatives files and visualization files for positive output level (the option `-o [int]`). The default value is 0 and in this case none of these files are written.

4.2 Data file

The data file is a simple ascii file with the extension **.metrix**. The options or data are set by means of keywords and values associated with those keywords (designed by [int], [float] or [type] for an integer value, a float value or a specific type if required, respectively). When an option is followed by {type}, it means that this option reads a list of values of this type (and not only one value like in [type]). They are generic options and options gathered by (included in) groups defined by a keyword which contains all the options related to a module.

There is no order for the list of keywords, but they are generally sorted by themes for readability purposes. The parser of the data file is not case sensitive for the keywords. For instance, `MetricType` is equivalent to `meTriCtyPe`. Moreover, the sign # is used to start a comment line.

We provide below the list of all the data classified by keywords group.

Isotropic : output an isotropic metric. The computed anisotropic metric is transformed into an isotropic one.

Anisotropic : enforce the output of an anisotropic metric. This is the default mode.

Fields {int} : specify the list of solution fields indices that are taken into account (*i.e.,* activated) for the metric computation. A metric is computed for each field. The resulting final metric is the intersection of those metrics. `Metrix` automatically computes the number of available fields in the solution file. By default, all the fields are active.

MetricType [type] or {type} : specify the chosen error estimate. If the same error estimate is used for all fields, only one metric type is set thanks to [**type**]. Otherwise, a metric type is provided for each field with the list of metric types **{type}**. The possible **type** choices for the error estimate are:

- **Hmet** for the interpolation error estimate
- **Gmet** for the iso-lines error estimate
- **Lmet** for the interface error estimate. The **Lmet** metric type must be supplied with some parameters associated with the interface error estimate. These parameters are the mesh size normal to the interface and the thickness of the interface refined region. This is prescribed by the syntax **Lmet [size,thick]** where **size** and **thick** are float values. These values must be put inside the brackets. For instance, **Lmet [0.1,0.02]**

Notice that the use of the goal oriented error estimate is not specified with **MetricType**. The goal oriented estimate is called and parametrized with the keyword **Adjoint**.

Norm [type] or {type} : specify the norm in which the error is controlled for all active fields thanks to [**type**] or for each field individually thanks to **{type}**. The possible **type** choices for the norm are **L1, L2, Linf** and more generally **Li**, with $i \in \mathbb{N}^+$.

TargetComplexity [int] or {int} : specify the desired mesh complexity. It is the theoretical continuous equivalent of the number of vertices. [**int**] specifies the same complexity for all fields. **{int}** specifies a list of complexities, one for each field. Cannot be used with the keyword **TargetError**.

TargetError [float] or {float} : specify the desired error threshold. It is mandatory for unsteady problems for consistency reason. [**int**] specifies the same error for all fields. **{int}** specifies a list of errors, one for each field. Cannot be used with the keyword **TargetComplexity**.

Derivatives [type] or {type} : specify the derivatives reconstruction method. If the same method is used for all fields, only one derivatives reconstruction type is set thanks to [**type**]. Otherwise, a derivatives reconstruction type is provided for each field with the list of types **{type}**. Two derivatives reconstruction methods are available:

- **L2Proj** the $\mathbf{L}^2$-projection derivatives reconstruction
- **Green** the Green formula based derivatives reconstruction.

MinSize [**float**] **:** specify the minimal authorized size prescribed by the metric.

MaxSize [**float**] **:** specify the maximal authorized size prescribed by the metric.

MaximalComplexity [**int**] **:** specify the maximal complexity authorized for the final metric. This option is used to limit the mesh size when **TargetError** is used or when several fields are considered. In the latter case, the resulting intersected metric may have a large complexity.

SkipRef [**int**] **{int} :** specify the list of references of triangles in 3D (edges in 2D) that are not considered as boundary elements. Indeed, `Metrix` assumes that a triangle with a reference is a boundary element and thus there is only one tetrahedron sharing this face in its internal algorithms. In the **SkipRef** option, the first [**int**] specifies the number of references to be skipped. Then, the list of these references is given thanks to **{int}**. For instance,

```
SkipRef   ←  this is the keyword
3         ←  this is the number of references to skip
2 3 56    ←  this is all the skipped references indices
```

GoalOriented : this keyword is used to activate the goal oriented error estimate. It also enables the user to set specific options for this error estimate until the next keyword. Notice that all the input files required for the goal oriented estimate are specified within this keyword and not with the `-in` or `-sol` command line options. After this keyword, the parameters associated with this error estimate are prescribed with the following syntax:

- **GOComplexity** [**int**] specify the desired mesh complexity for the goal oriented error estimate. If it is not specified, the complexity associated with the first field is used. Cannot be used with the keyword **AdjError** [**float**].
- **GOError** [**float**] specify the desired error threshold for the goal oriented error estimate. If it is not specified, the error threshold associated with the first field is used. Cannot be used with the keyword **AdjComplexity** [**int**].
- **GODerivatives** [**type**] specify the derivatives reconstruction method for the goal oriented error estimate. Two derivatives reconstruction methods are available:
  - **L2Proj** the $\mathbf{L}^2$-projection derivatives reconstruction
  - **Green** the Green formula based derivatives reconstruction.

  If it is not specified, the derivatives reconstruction method of the first field is used.
- **Files** [**char1**] [**char2**] specify the files to read containing the fields for the goal oriented error estimate. [**char1**] corresponds to the file containing the **variables** for the estimate. [**char2**] corresponds to the file containing the **weights** for the estimate.
  A list of files can be prescribed to `Metrix` (a maximum of 20 couples variable/weight).

Practically, the goal oriented estimate is given by the control of the interpolation error in $\mathbf{L}^1$ norm of a sum of variables weighted by some fields. For this reason, two files are associated to evaluate the error estimate. See [7] for technical details.

Gradation : this keyword is used to activate the mesh gradation and to specify that the following options until the next keyword are associated with the gradation module. The mesh gradation can be used for Module 1 and 5. After this keyword, the parameters associated with this module are prescribed with the following syntax:

- **SizeGradation [float]** : maximal size variation coefficient between two neighboring vertices. This coefficient must be greater than 1.
- **EdgeBased** : the metric gradation algorithm is performed edge by edge.
- **SuperEdgeBased** : the metric gradation algorithm is performed super-edge by super-edge.
- **PhysicalSpace** : the gradation is homogeneous in the physical space. This is the default mode.
- **MetricSpace** : the gradation is homogeneous in the metric space.
- **MixedSpace** : the gradation is homogeneous in a mixed physical/metric space defined by the mixed space coefficient.
- **MixedSpaceCoefficient [float]** : this coefficient defines the mixed physical/metric space. This coefficient lies in $[0, 1]$. Default value is 0.125.
- **HLinear** : linear variation law on the metric size (h).
- **HGeometric** : geometric variation law on the metric size (h).
- **LLinear** : linear variation law on the metric eigenvalues (λ).
- **HinvLinear** : linear variation law on the inverse of the metric size (h^{-1}).

The complete description of all these options and their technical properties is given in [2].

UserMetric : this keyword is used to activate the user metric module and to specify that the following options until the next keyword are associated with the user metric module. This module is used in Module 1 and 7. After this keyword, the parameters associated with this module are prescribed with the following syntax:

- **SizeCoef [float]** the normal size coefficient. The normal size at each vertex is the minimal size in the tangent plane times this coefficient. Default value is 1. Cannot be used with the option **NormalSize [float]**.

- **NormalSize** [**float**] the normal size of the metric defined on the surface. Cannot be used with **SizeCoef** [**float**].
- **UserMetricSizeGradation** [**float**] maximal size variation coefficient between two neighboring vertices. This coefficient is only used for the construction of the user metric. It can be different from the standard size gradation. This coefficient must be greater than 1.

Now, we give an example of data file

```
## DEFAULT FILES FOR METRIX
## Generated by METRIX
Anisotropic

# Fields : Specifies solution fields indices to take into account
Fields
 1

# Norm : L1, L2, Linf or Li where i is > 0
Norm
 L2

# Metric Type : Hmet = Hessian, Gmet = Gradiant matrix, Lmet = Level Set
MetricType
 Hmet
# For level set size and thickness by using Lmet [2,0.01]

## TargetError : Specifies desired target errors
TargetError
  0.01

## TargetComplexity : if you prefer specify the complexity (nbr of vertices)
## instead of TargetError
# TargetComplexity
# 1000

## Hessian : L2Proj, Green
# Derivatives
# L2Proj

# Metric gradation control
Gradation
  SizeGradation 1.8
```

```
  EdgeBased
  MetricSpace
  HLinear

## GoalOriented : specify the goal-oriented adaptation parameters
## Here the target error and the hessian are the same as field 1
#GoalOriented
#  Files fluxx.solb djdx.solb
#  Files fluxy.solb djdy.solb
#  Files fluxz.solb djdz.solb

#UserMetric
#  GeomMinSize # OR GeomMeanSize
#  SizeCoef  1.000000
```

5 Some application examples

We present some results obtained with `Metrix`. In two dimensions, meshes have been generated with `Yams` [4]. And, for the three dimensional case, two mesh generators have been considered `Gamic` [5] and `Mmg3d` [3].

5.1 Multi-scales mesh adaptation

In order to illustrate the multi-scales mesh adaptation, we consider the case of a supersonic flow simulation around the supersonic business jet geometry (SSBJ) provided by Dassault-Aviation. The jet is flying at cruise with Mach number 1.6, an angle of attack of 3° and an altitude of 45,000 feet.

The aircraft length is 30 metres and the mesh size on the aircraft geometry is between 1 millimetre to 30 centimetres, Figure 1 (left). This represents already a size variation of four orders of magnitude with respect to the aircraft size, and this only for the surface mesh. The computational domain is a cylinder of 2.5 kilometers length and 2.5 kilometers diameter, Figure 1 (right). This represents a scale factor of $2.5\,10^6$ if the size of the domain is compared to the maximal accuracy of the aircraft surface mesh. These large scale factors will illustrate the multi-scales property of the mesh adaptation.

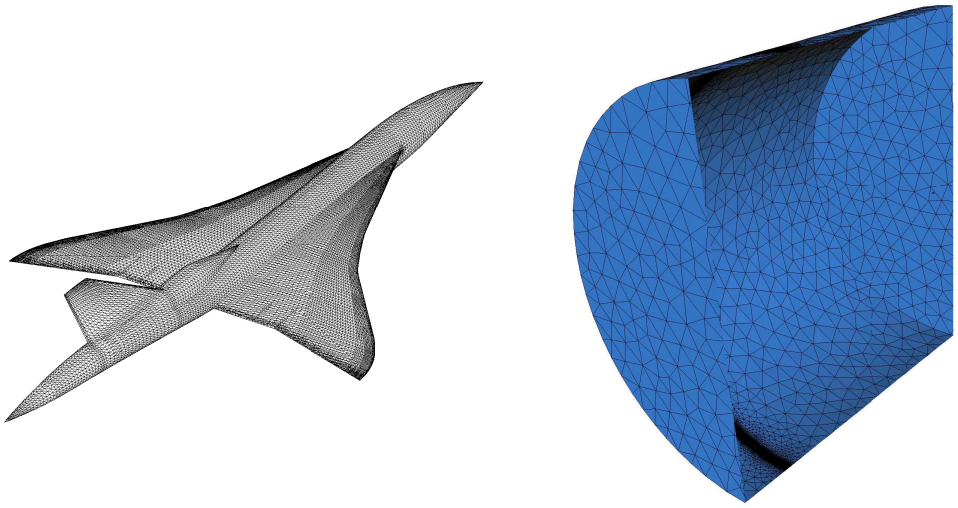


Figure 1: Left, surface mesh of the supersonic business jet geometry. Right, the two kilometres and a half cylindrical computational domain.

As regards mesh adaptation, we choose to control in norm $\mathbf{L}^2$ the Mach number variable. 20 mesh adaptation iterations have been performed and we use the Green formula based Hessian reconstruction. Finally, we specify a complexity leading to a final adapted mesh containing 3 764 591 vertices and 22 324 258 tetrahedra. This mesh is illustrated in Figures 2 and 3.

Supersonic flows involve highly anisotropic physical structures. In the case of a supersonic aircraft, a large number of shock structures are present in the flow that correspond to all Mach cones issued from the aircraft geometry, Figure 3 (bottom). The final adapted mesh is highly anisotropic, cf. Figure 2. Figure 3 shows that refinements along Mach cones have been propagated in the whole computational domain (top left) with a high accuracy (or small size) in shock regions (middle left) despite the large scale factor between the mesh accuracy and the domain size.

These anisotropic meshes reduce considerably the number of vertices required to get this accuracy and furthermore they also reduce drastically the flow solver numerical diffusion allowing shock waves to be very accurately propagated in the whole computational domain, Figure 3 (bottom).

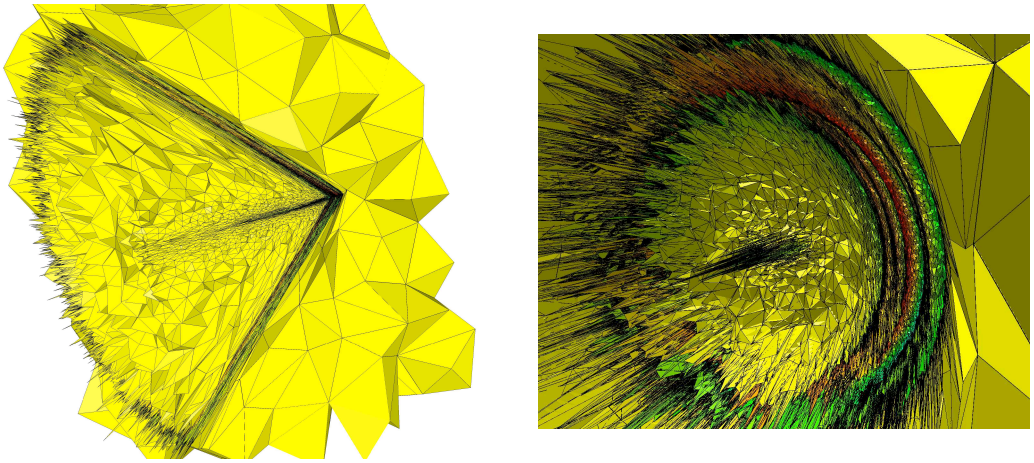


Figure 2: 3D supersonic aircraft simulation. Two views of the final anisotropic adapted mesh containing $3,764,591$ vertices and $22,324,258$ tetrahedra throughout cut planes. The global (left) and the close-up views point out the highly anisotropic feature of the mesh.

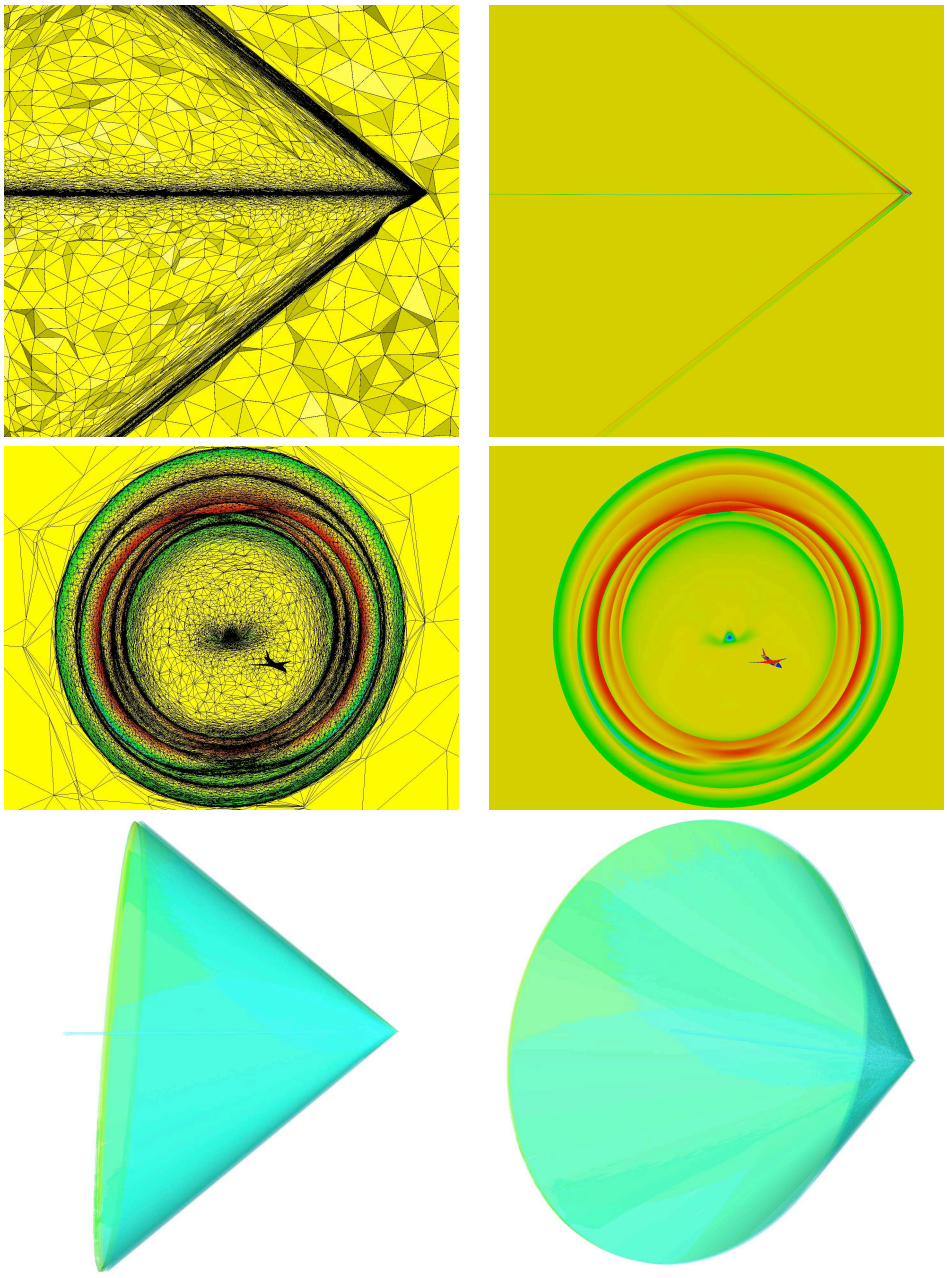


Figure 3: 3D supersonic aircraft simulation. Two views of the final anisotropic adapted mesh throughout cut planes (left) and the associated Mach number iso-values (right). Top, the cut plane is the symmetry plane. Middle, the cut plane is behind the aircraft orthogonal to the jet path. Bottom, Mach number iso-surface representing the Mach cones issued from the aircraft in the whole domain.

5.2 Goal-oriented mesh adaptation

We re-consider the flow around a complete aircraft geometry flying at cruise speed Mach 1.6 with an angle of attack of 3°. The aim is to compute accurately the pressure 100 meters below the aircraft in a plane parallel to the flow, see Figure 4 (left). We denote by γ this region which will be the observation zone for the goal-oriented mesh adaptation. This region is defined by:

$$\gamma = \left\{ (x,y,z) \in \mathbb{R}^3 \;\middle|\; 100 \leq x \leq 140\,,\;\; -1 \leq y \leq 1\,,\;\; z = 100 \right\}.$$

We compare the multi-scales mesh adaptation strategy [7] with the goal-oriented mesh adaptation [7]. The multi-scales approach considers an adaptation on the local Mach number in $\mathbf{L}^2$ norm. The functional for the goal-oriented method is

$$j(\gamma, W) = \int_\gamma \left(\frac{p - p_\infty}{p_\infty} \right)^2 \mathrm{d}\gamma\,.$$

In both cases, the Hessian is reconstructed with the Green formula method. A complexity of 160 000 has been set for the final adaptation. The final meshes with both methods are composed of almost 800 000 vertices.

The differences in the mesh refinement are clearly highlighted in Figure 5. The multi-scales adaptation adapts the mesh in the whole domain to control the global error on the local Mach number, while the goal-oriented strategy focalizes its adaptation on the observation region. The gain in accuracy is pointed out in the right picture of Figure 4.

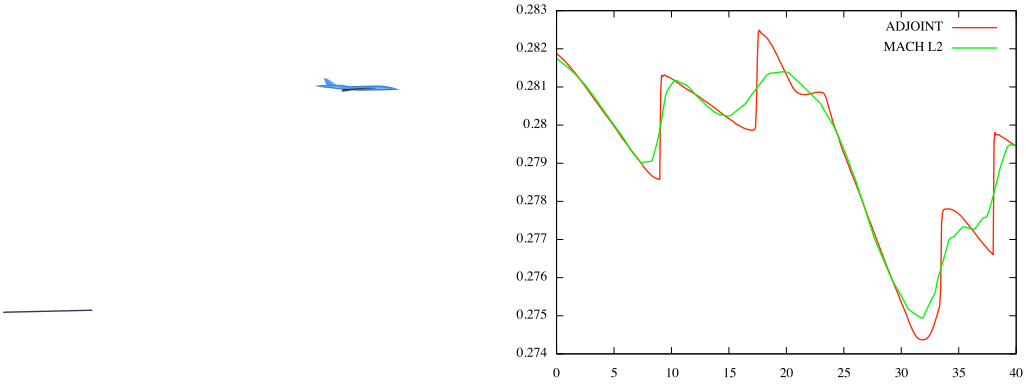


Figure 4: *3D supersonic aircraft simulation. Left, position of the observation plane with respect to the aircraft. Right, comparison of the pressure in the observation region between the goal-oriented (red) and the multi-scales (green) mesh adaptations for the same mesh size.*

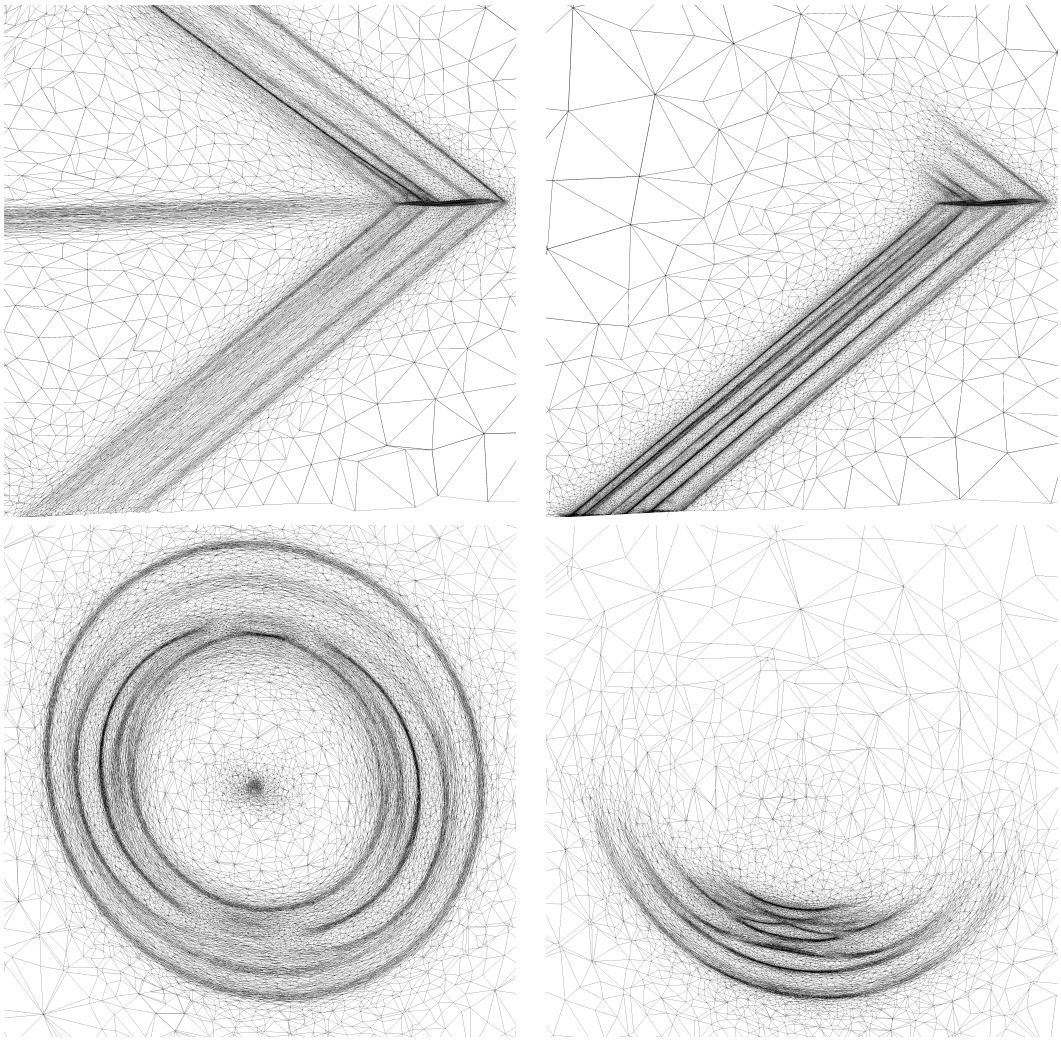


Figure 5: *3D supersonic aircraft simulation. Left, the final anisotropic mesh with the multi-scales mesh adaptation in* $\mathbf{L}^2$ *norm. Right, the final anisotropic mesh with the goal-oriented mesh adaptation.*

5.3 Mesh adaptation for bi-fluid flows

In the context of the simulation of bi-fluid flows or more generally multi-fluid flows, the interface can be modeled by the level set method. In this case, a level set function is considered to represent the location of the interface separating two non-miscible fluids, for instance air and water. Formally speaking, let ϕ be a level set function, $\Phi_0 = \{\mathbf{x} \,|\, \phi(\mathbf{x}) = 0\}$ describes the interface and $\phi(\mathbf{y})$ is the distance of $\mathbf{y}$ to the interface. Moreover, by definition the function ϕ verifies: $\nabla\phi = 1$ and $\mathbf{n}_\phi = \nabla\phi$. In the context of level set applications, we aim at meshing accurately the interface between the two fluids and at coarsening the mesh as we gradually get further from the interface. We can define a metric for the interface, *i.e.,* Φ_0, of the form [6, 7]:

$$\mathcal{M}_{\Phi_0} = {}^t(\nabla\phi\ \nabla\phi^\perp) \begin{pmatrix} h_{user}^{-2} & 0 \\ 0 & \varepsilon^{-2} \end{pmatrix} (\nabla\phi\ \nabla\phi^\perp)\,,$$

where h_{user} is the size in the direction normal to Φ_0 prescribed by the user and ε is the size in the direction tangent to Φ_0 which depends on a specific error model, for instance the curvature of Φ_0. Then, the mesh gradation enables a smooth metric field to be determined continuously in the overall domain.

Two bi-fluids examples studied in [6] are presented. Figure 6 displays the use of this metric for a two-dimensional simulation of a breaking water column which impacts an obstacle. The level set function is shown on the left and the adapted mesh generated from the interface metric on the right. The wave interface impacting the obstacle is clearly visible inside the mesh. The three-dimensional case is shown in Figure 7. The complexity of the interface is illustrated in the top picture, it represents a breaking wave with several tubes. The adapted surface and volume meshes are displayed in the middle and bottom pictures.

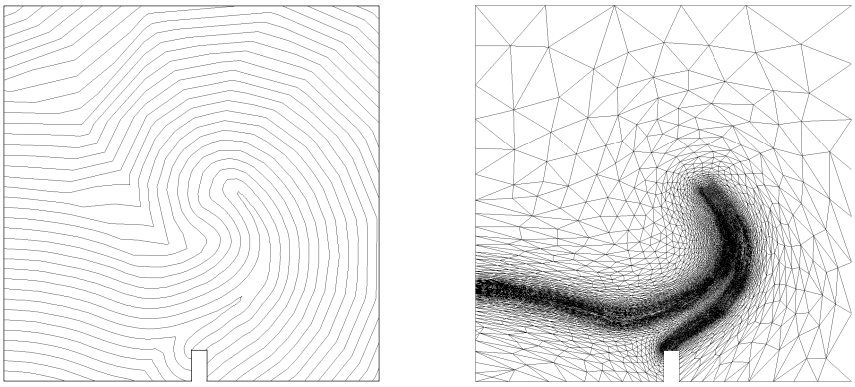


Figure 6: *Application of the interface error estimate to a 2D bi-fluid simulation. Left, the level set function of a water wave impacting an obstacle. Right, the adapted mesh generated from the level set metric. The interface is clearly visible inside the mesh.*

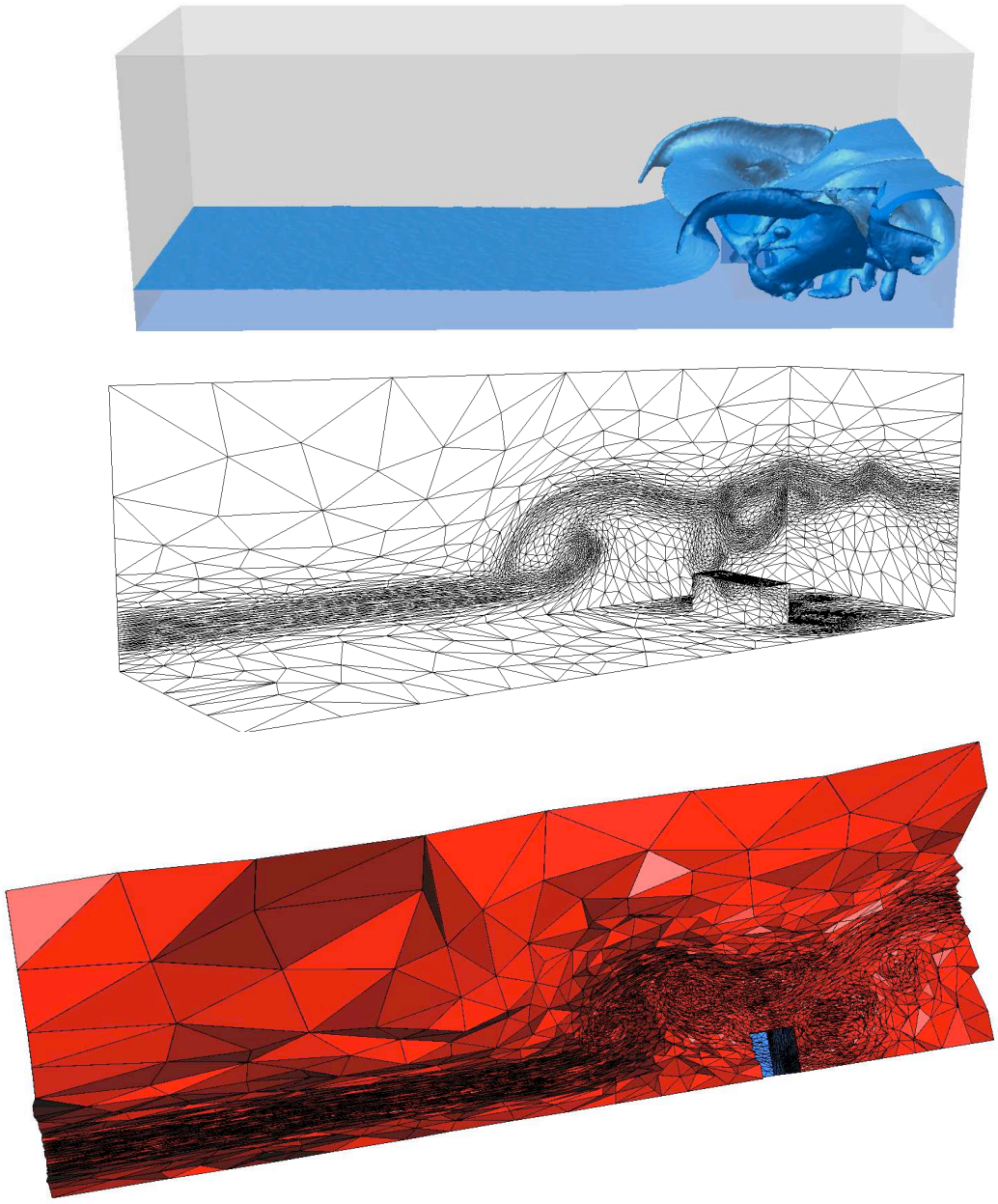


Figure 7: *Application of the interface error estimate to a 3D bi-fluid simulation. Up, interface represented by the zeros of the level set function. Middle, the adapted surface mesh, and down, the adapted volume mesh generated from the level set metric. The interface is clearly visible inside the mesh.*

5.4 Mesh gradation control

The mesh gradation module (Module 5) is presented in a two-dimensional examples. The edge-based algorithm and the h-linear variation law have been considered. The homogeneous gradation in the metric space, named **method 1**, is compared to the homogenous one in the physical space, called **method 2**. A mesh adaptation with no gradation has been also done (in fact we have prescribed a gradation of 10^{12}). Table 1 summarizes all the performed cases and the meshes sizes.

The scramjet configuration at Mach 3 is typical of numerical simulations in compressible fluids involving highly anisotropic phenomena with very strong shocks. The solution of this simulation results in a complex flow with several shock reflexions and shocks that cross over. However, all these shock waves have simple straight line shapes. This flow solution leads to a metric field prescribing large aspect ratio elements in shock region with a very small size prescription in the direction orthogonal to the shock, and prescribing large sizes where the solution is smooth or constant. It outcomes a particularly non regular metric field.

First of all, the adapted mesh with no gradation is depicted in Figure 9. We observe that the mesh is of poor quality for a numerical computation and that the mesh generator has a lot of difficulties to generate elongated elements in shock regions. The mesh lacks of resolution in shock region. This is due to the impossibility to generate a unit mesh in the given non regular metric field. In particular, in the neighboring region of a shock, the computed metric is highly dependent of the vertex position. If the vertex is inside the shock then an appropriate metric is computed, but if the vertex is slightly outside the shock, the information is lost.

The meshes obtained with method 1 and method 2 are presented in Figures 10 and 11. In each case shock regions are meshed anisotropically with good quality as compared to the case with no mesh gradation. The anisotropy is preserved and a high resolution is obtained. However, the results of the two methods are particularly different, even if for the same gradation parameter they provide meshes of almost the same size. The elements become isotropic when growing with the method 2, whereas they remain anisotropic along the shock waves with method 1. Method 2 provides moderate anisotropic meshes as compared to method 1.

In conclusion, for such simulation, the mesh gradation is mandatory to generate high-quality anisotropic meshes.

	$\beta = 1.3$	$\beta = 1.5$	$\beta = 2$	$\beta = 10^{12}$
Metric-space-homogeneous gradation	$51,456$	$33,754$	$21,908$	$8,959$
Physical-space-homogeneous gradation	$50,538$	$32,991$	$21,841$	-

Table 1: Number of vertices of the adapted mesh in each case for the scramjet simulation.

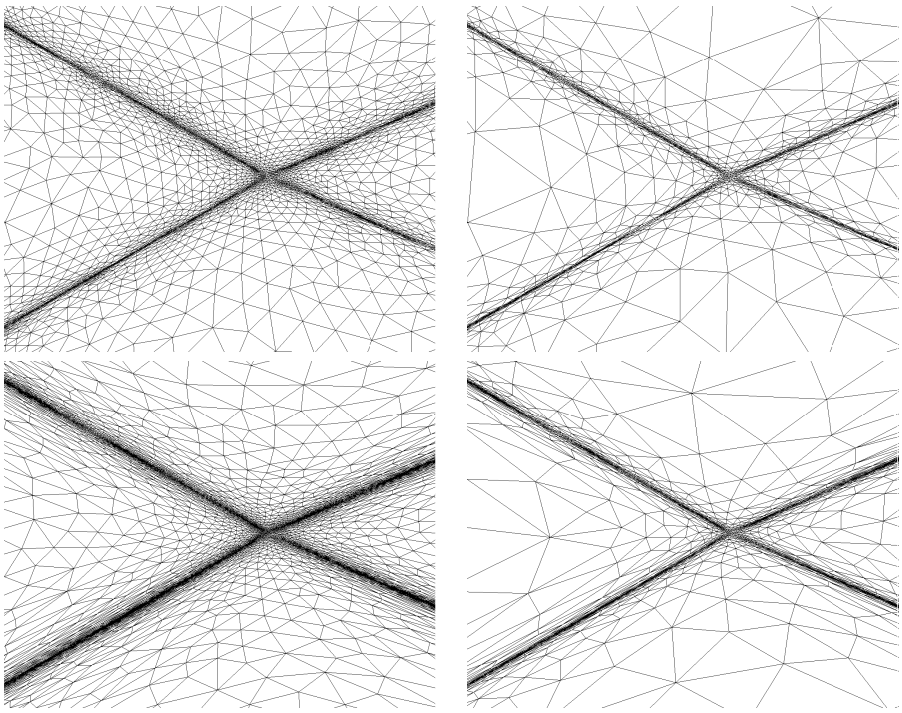


Figure 8: *Adapted meshes resulting from the simulation of a scramjet with the physical-space-homogenous (top) and the metric-space-homogeneous (bottom) gradation. This zoom on the mesh shows that the anisotropy has been preserved in each case. Left, a gradation coefficients of* 1.3 *and right a value of* 2*.*

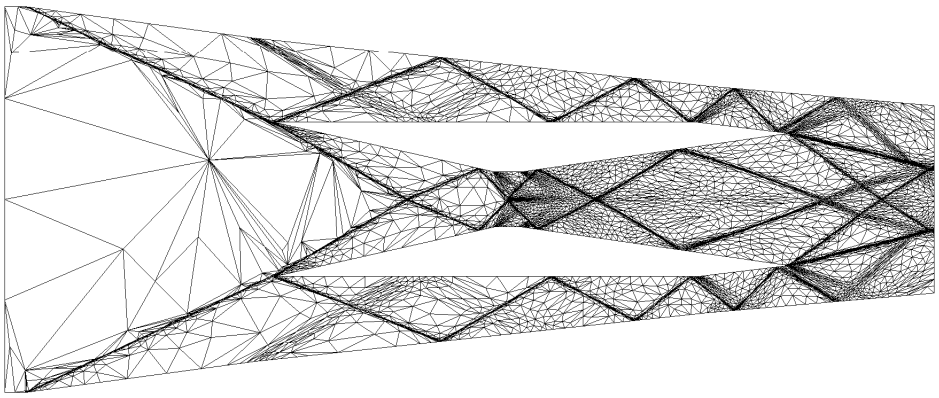


Figure 9: *Adapted mesh resulting from the simulation of a scramjet with no mesh gradation.*

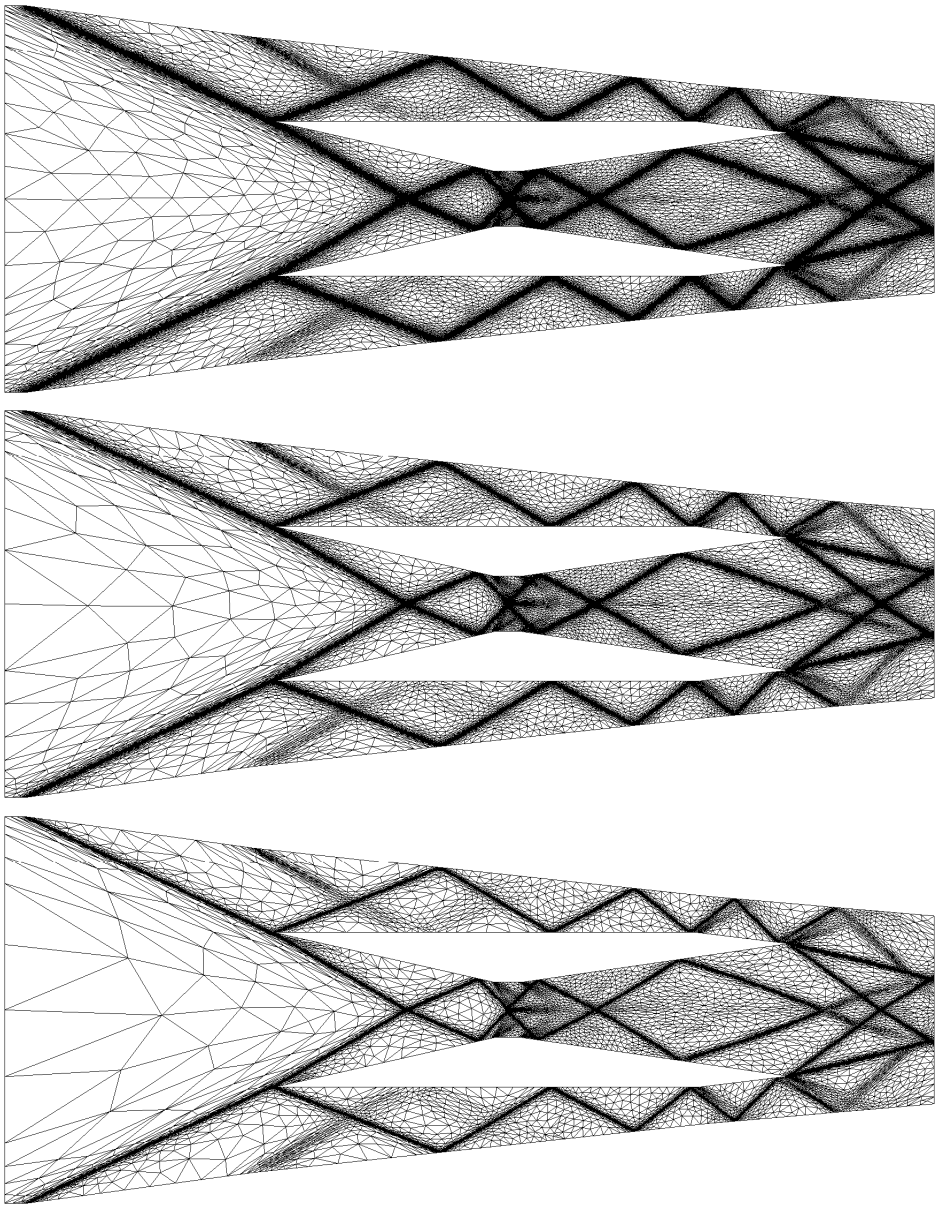


Figure 10: *Adapted meshes resulting from the simulation of a scramjet with the metric-space-homogeneous gradation. From top to bottom, gradation coefficients of* 1.3, 1.5 *and* 2 *have been used.*

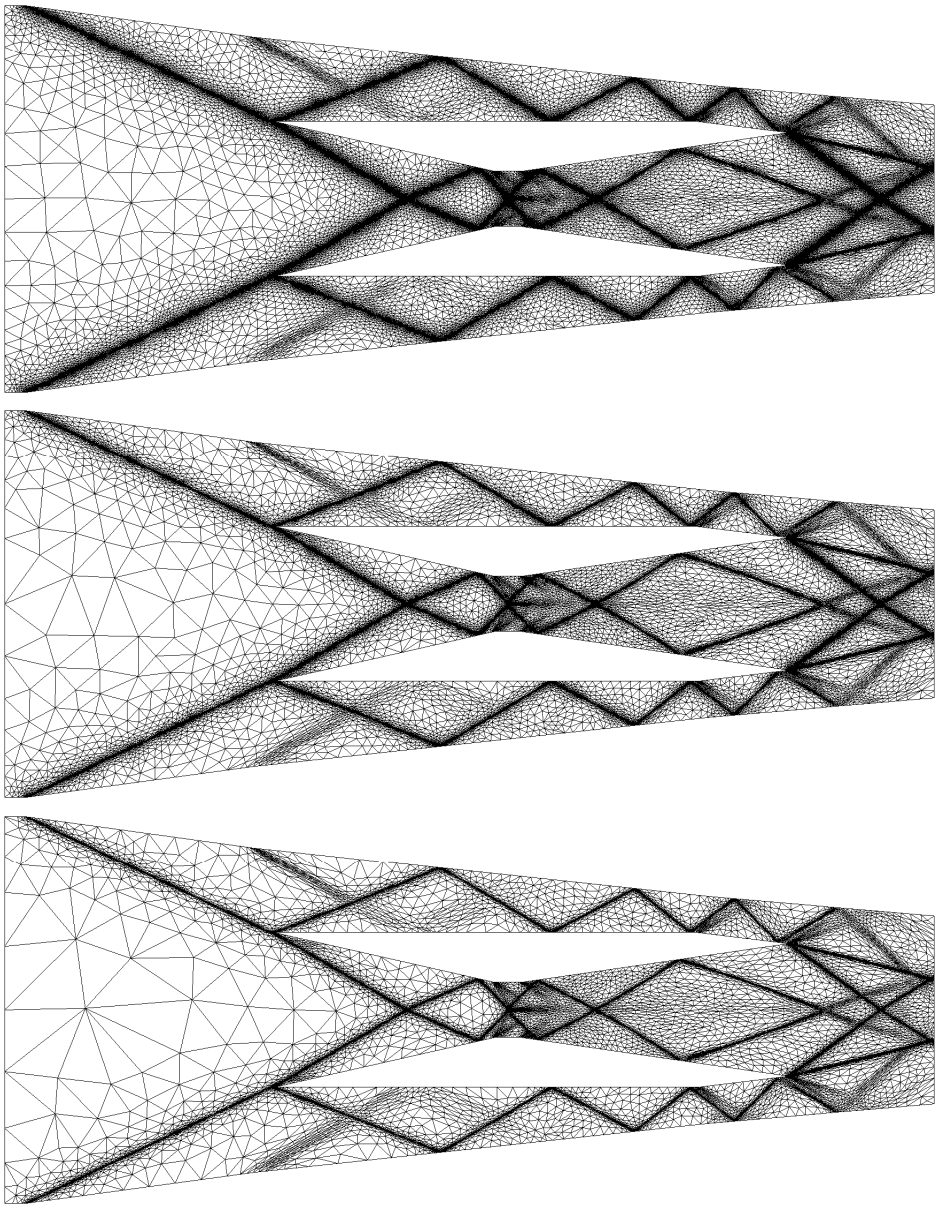


Figure 11: *Adapted meshes resulting from the simulation of a scramjet with the physical-space-homogeneous gradation. From top to bottom, gradation coefficients of* 1.3, 1.5 *and* 2 *have been used.*

5.5 Mesh control with the user metric

The user metric module (Module 7) can be used for the generation of a well-graded quasi-uniform volume mesh around a geometry. The unique available data to define the metric is the surface mesh, *i.e.,* the mesh of the boundary. The volume mesh accuracy close to the geometry is managed by the normal size specification and by the gradation parameter. These data enable us to define a metric field in the entire domain.

First the metric at each vertex of the surface mesh is computed. To this end, for each surface triangle K, we evaluate in the triangle plane the unique 2D metric for which this triangle is unit:

$$\overline{\mathcal{M}}_K = {}^t(\bar{\mathbf{t}}_\mathbf{1}\, \bar{\mathbf{t}}_\mathbf{2}) \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} (\bar{\mathbf{t}}_\mathbf{1}\, \bar{\mathbf{t}}_\mathbf{2})\,.$$

Then, the size on the triangle normal $\mathbf{n}$ is set through the user parameter. For instance, it can be a fixed size $h_n = h_{fix}$ or $h_n = \alpha \min(\lambda_1^{-1/2}, \lambda_2^{-1/2})$ where α is prescribed. It results in a 3D metric for each element of the surface mesh:

$$\mathcal{M}_K = {}^t(\mathbf{t_1}\, \mathbf{t_2}\, \mathbf{n}) \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & h_n^{-2} \end{pmatrix} (\mathbf{t_1}\, \mathbf{t_2}\, \mathbf{n})\,.$$

The metric at each vertex $\mathbf{p}$ of the surface is obtained by a local $\mathbf{L}^2$-projection in the log-Euclidean framework:

$$\mathcal{M}_p = \exp\left(\frac{\displaystyle\sum_{K_i \ni P} |K_i| \log(\mathcal{M}_{K_i})}{\displaystyle\sum_{K_i \ni P} |K_i|} \right)\,.$$

This metric, only defined on the surface, is propagated in the entire domain thanks to the mesh gradation procedure.

The generation of a well-graded quasi-uniform mesh is illustrated on a three-dimensional example. We consider an aircraft in a spherical domain and we aim at generating an isotropic graded mesh with a Delaunay-based adaptive mesh generator [5]. We specify a normal size coefficient equal to 1 and a size gradation of 1.2 and 1.5. We obtain meshes made up of $3,574,143$ and $946,236$ tetrahedra, respectively. These meshes are shown in Figure 12.

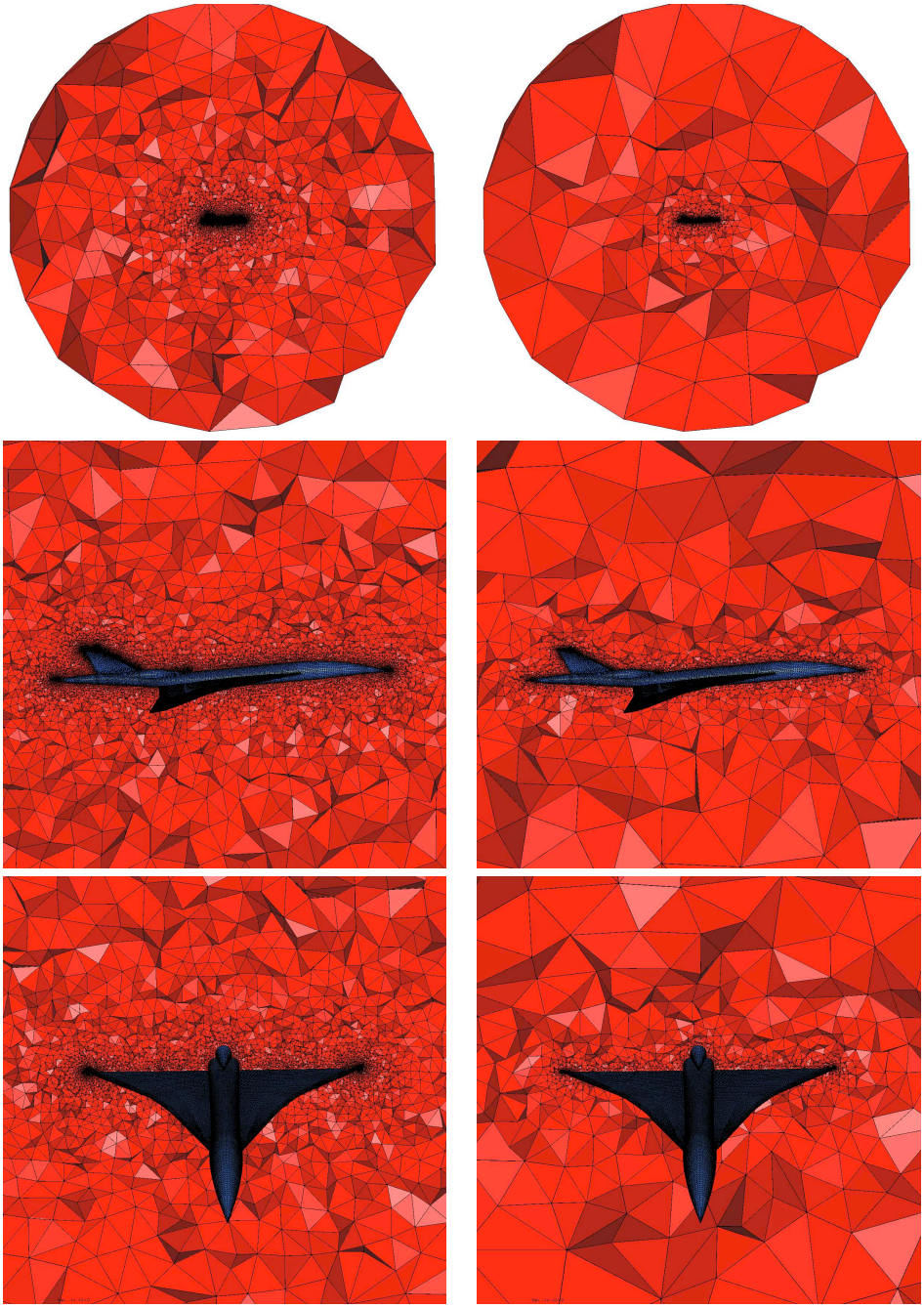


Figure 12: *Graded isotropic mesh of an aircraft in a spherical domain. Left, the mesh for a size gradation of* 1.2 *containing* $3,574,143$ *tetrahedra. Right, the mesh for a size gradation of* 1.5 *containing* $946,236$ *tetrahedra.*

References

[1] F. Alauzet. *Adaptation de maillage anisotrope en trois dimensions. Application aux simulations instationnaires en Mécanique des Fluides*. PhD thesis, Université Montpellier II, Montpellier, France, 2003. (in French).

[2] F. Alauzet. Size gradation control of anisotropic meshes. *Finite Elements in Analysis and Design*, 2009. to appear.

[3] C. Dobrzynski and P.J. Frey. Anisotropic delaunay mesh adaptation for unsteady simulations. In *Proc. of 17th Int. Meshing Rountable*, pages 177–194. Springer, 2008.

[4] P.J. Frey. `Yams`, a fully automatic adaptive isotropic surface remeshing procedure. RT-0252, INRIA, November 2001.

[5] P.-L. George. Tet meshing: construction, optimization and adaptation. In *Proc. of 8th Int. Meshing Rountable*, South Lake Tao, CA, USA, 1999.

[6] D. Guegan. *Modélisation numérique d'écoulements bifluides 3D instationnaires avec adaptation de maillage*. PhD thesis, Université de Nice Sophia Antipolis, Sophia Antipolis, France, 2007. (in French).

[7] A. Loseille. *Adaptation de maillage anisotrope 3D multi-échelles et ciblée à une fonctionnelle pour la mécanique des fluides. Application à la prédiction haute-fidélité du bang sonique*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2008. (in French).

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803